UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Regresión no Lineal

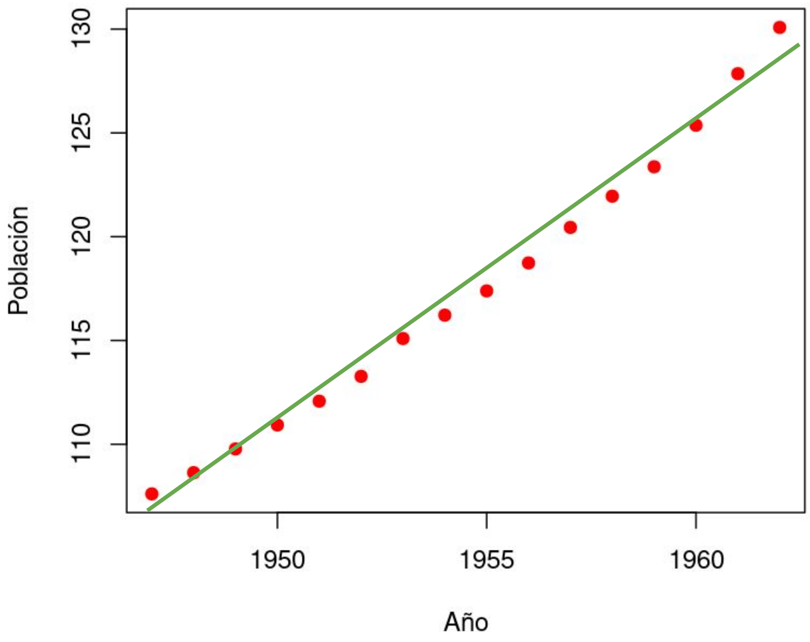
Cristian López Del Alamo
clopezd@utec.edu.pe
IPRODAM3D - Research group

2022

1 Recordando: Regresión no Lineal

Objetivo: En esta clase es entender la similitud entre la regresión lineal y la regresión no lineal, las aplicaciones y el algoritmo de aprendizaje de máquinas que está relacionado.

Regresión Lineal Simple



Hipótesis

$$h(x_i) = b + wx_i$$

Función de Pérdida

$$\mathcal{L} = \frac{\sum_{i=0}^{n}(y_i - h(x_i))^2}{2n}$$

Cálculo de Derivadas

$$\frac{\partial \mathcal{L}}{\partial w} = \frac{\sum_{i=0}^{n}(y_i - h(x_i))(-x_i)}{n}$$

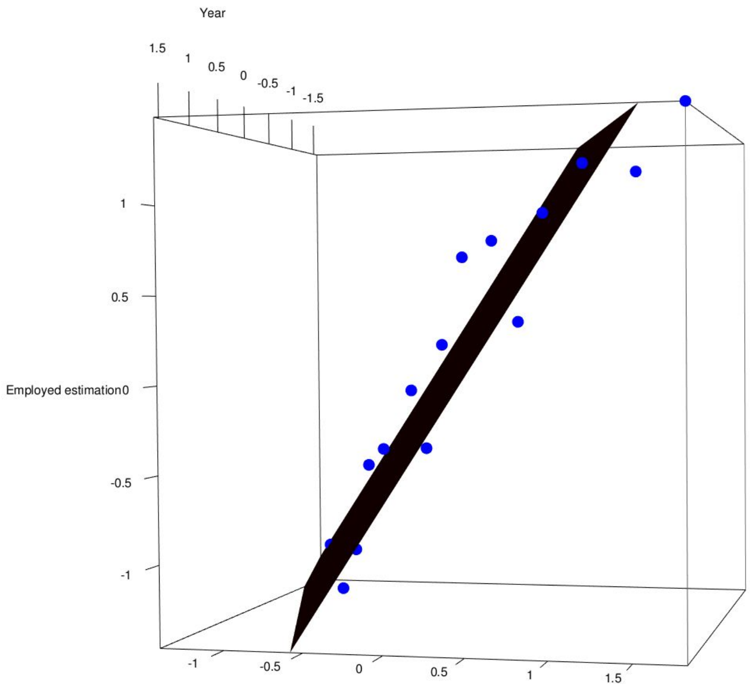
$$\frac{\partial \mathcal{L}}{\partial b} = \frac{\sum_{i=0}^{n}(y_i - h(x_i))(-1)}{n}$$

Actualización

$$w = w - \alpha\frac{\partial \mathcal{L}}{\partial w}$$

$$b = b - \alpha\frac{\partial \mathcal{L}}{\partial b}$$

Regresión lineal múltiple: Caso 2 variables



Hipótesis

$$h(x_i) = b + x_{(i,1)}w_1 + x_{(i,2)}w_2$$

Función de Pérdida

$$\mathcal{L} = \frac{\sum_{i=0}^{n}(y_i - h(x_i))^2}{2n}$$

Cálculo de Derivadas

$$\frac{\partial \mathcal{L}}{\partial w_j} = \frac{\sum_{i=0}^{n}(y_i - h(x_i))(-x_{(i,j)})}{n}$$

$$\frac{\partial \mathcal{L}}{\partial b} = \frac{\sum_{i=0}^{n}(y_i - h(x_i))(-1)}{n}$$

Actualización

$$w_j = w_j - \alpha\frac{\partial \mathcal{L}}{\partial w_j}$$

$$b = b - \alpha\frac{\partial \mathcal{L}}{\partial b}$$

Regresión lineal múltiple: Caso k variables

Hipótesis

$$h(x_i) = b + x_{(i,1)}w_1 + x_{(i,2)}w_2 + ... + x_{(i,k)}w_k$$

$$h(x_i) = 1w_0 + x_{(i,1)}w_1 + x_{(i,2)}w_2 + ... + x_{(i,k)}w_k$$

$$h(x_i) = x_{(i,0)}w_0 + x_{(i,1)}w_1 + x_{(i,2)}w_2 + ... + x_{(i,k)}w_k$$

$$h(x_i) = [x_{(i,0)} \quad x_{(i,1)} \quad x_{(i,2)} \quad ... \quad x_{(i,k)}][w_0 \quad w_1 \quad w_2 \quad ... \quad w_k]^T$$

$$\boxed{h(x_i) = x_i * w^T}$$

Regresión lineal múltiple: Caso k variables

Hipótesis

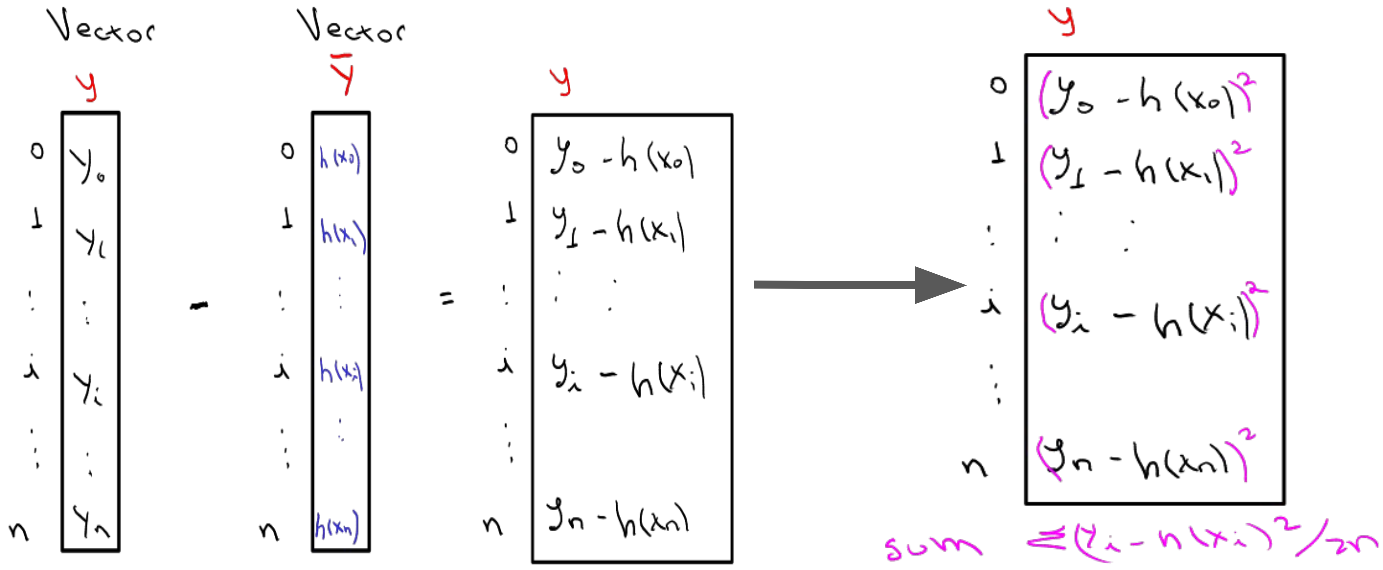
Matrix X Vector w Vector $\bar{Y}$

$$
\begin{array}{c|cccc|}
 & 0 & 1 & \cdots & k \\
\hline
0 & 1 & x_{01} & \cdots & x_{0k} \\
1 & 1 & x_{11} & \cdots & x_{1k} \\
\vdots & 1 & \cdots & \cdots & \cdots \\
i & 1 & x_{i1} & \cdots & x_{ik} \\
\vdots & 1 & \cdots & \cdots & \cdots \\
n & 1 & x_{n1} & \cdots & x_{nk} \\
\hline
\end{array}
*
\begin{array}{c|c|}
\hline
0 & W_0 \\
1 & W_1 \\
\vdots & \vdots \\
K & W_k \\
\hline
\end{array}
=
\begin{array}{c|c|}
\hline
0 & h(x_0) \\
1 & h(x_1) \\
\vdots & \vdots \\
i & h(x_i) \\
\vdots & \vdots \\
n & h(x_n) \\
\hline
\end{array}
$$

$$h(X) = X * w^T$$

Regresión lineal múltiple: Caso k variables

Función de Pérdida



$$\mathcal{L} = ||y - h(X)||_2^2$$

Regresión lineal múltiple: Caso k variables

Cálculo de derivadas

$$
\begin{array}{c|c|}
 & y - \bar{y} \\
0 & y_0 - h(x_0) \\
1 & y_1 - h(x_1) \\
\vdots & \vdots \quad \vdots \\
i & y_i - h(x_i) \\
\vdots & \\
n & y_n - h(x_n)
\end{array}
\qquad
\begin{array}{c|cccccc|}
 & & & & -X_j & & \\
 & 0 & 1 & \cdots & j & & k \\
0 & 1 & x_{01} & \cdots & -x_{0j} & & x_{0k} \\
1 & 1 & x_{11} & \cdots & -x_{1j} & & x_{1k} \\
\vdots & 1 & \cdots & \cdots & \cdots & & \cdots \\
i & 1 & x_{i1} & \cdots & -x_{ij} & & x_{ik} \\
\vdots & 1 & \cdots & \cdots & \cdots & & \cdots \\
n & 1 & x_{n1} & \cdots & -x_{nj} & & x_{nk}
\end{array}
$$

T

$$\frac{\partial \mathcal{L}}{\partial w} = \frac{(Y - h(X))^T * (-1 * X)}{n}$$

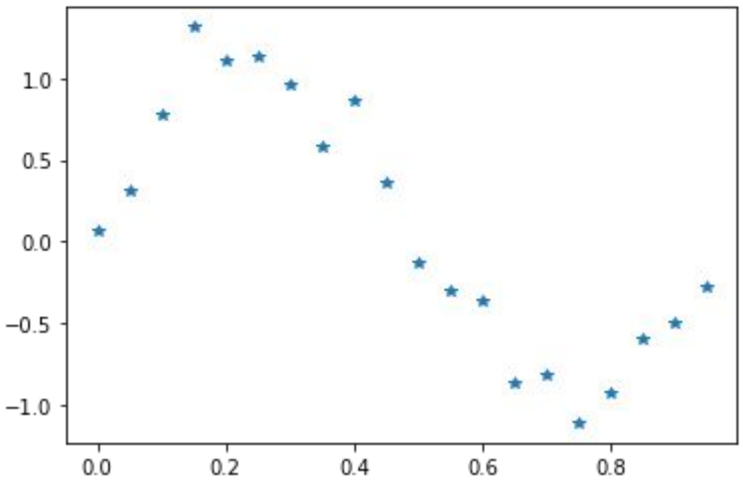
Regresión lineal múltiple: Caso k variables

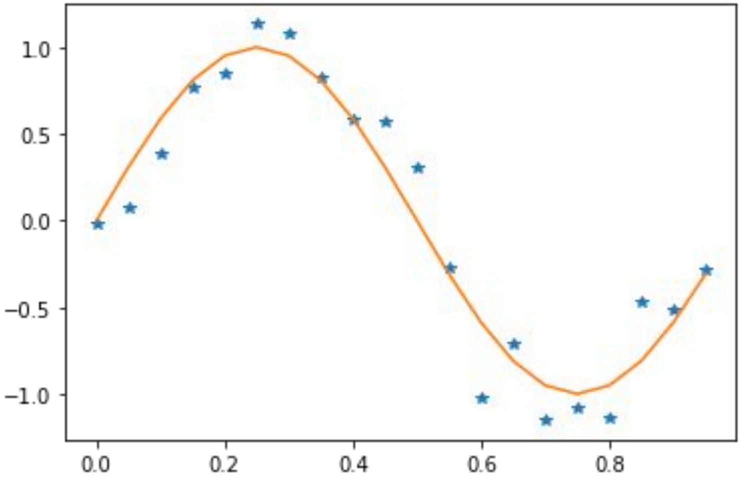
Actualizando w

$$W = W - \alpha * \frac{\partial \mathcal{L}}{\partial w}$$

2 Regresión no Lineal

Objetivo: Entender la matemática detrás de la regresión no lineal multivariable


1.0
0.5
0.0
−0.5
−1.0
0.0
0.2
0.4
0.6
0.8


1.0
0.5
0.0
−0.5
−1.0
0.0
0.2
0.4
0.6
0.8

UTEC UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Polinomio

hipótesis: $h(x_i) = b + x_i w_1 + x_i^2 w_2 + x_i^3 w_2 + \ldots + + x_i^p w_2$

hipótesis: $$h(x_i) = b + \sum_{j=1}^{p} x_i^j w_j$$

hipótesis: $$h(x_i) = x_i^0 w_0 + \sum_{j=1}^{p} x_i^j w_j$$

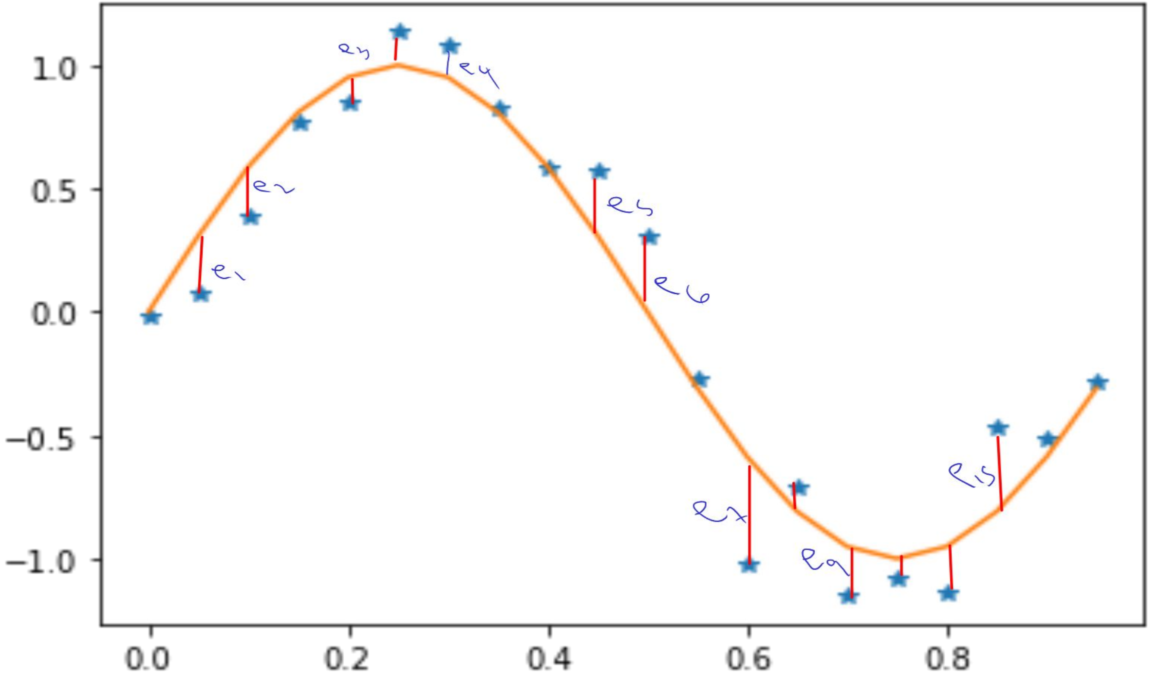
hipótesis: $$h(x_i) = \sum_{j=0}^{p} x_i^j w_j$$

$$X \cdot W^T = h(X) = \hat{Y}$$

$$\begin{bmatrix} \boxed{x_1^0 \quad x_1^1 \quad x_1^2 \quad \ldots \quad x_1^p} \\ x_2^0 \quad x_2^1 \quad x_2^2 \quad \quad x_2^p \\ \vdots \\ x_i^0 \quad x_i^1 \quad x_i^2 \quad \ldots \quad x_i^p \\ \vdots \\ x_n^0 \quad x_n^1 \quad x_n^2 \quad \ldots \quad x_n^p \end{bmatrix} \times \boxed{\begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ \vdots \\ w_p \end{bmatrix}} = \begin{bmatrix} \boxed{h(x_1)} \\ h(x_2) \\ \\ h(x_i) \\ \\ h(x_n) \end{bmatrix}$$

$$h(X) = X w^t$$

Función de Pérdida



Función de Pérdida

$$\mathcal{L} = \frac{\sum_{i=0}^{n}(y_i - h(x_i))^2}{2n}$$

$$
\begin{array}{ccccccc}
Y & - & \bar{Y} & = & Y - \bar{Y} & & \bar{e} \\
\begin{vmatrix} y_1 \\ y_2 \\ \vdots \\ y_i \\ \vdots \\ y_n \end{vmatrix} & - & \begin{vmatrix} \boxed{h(x_1)} \\ h(x_2) \\ \\ h(x_i) \\ \\ h(x_n) \end{vmatrix} & = & \begin{vmatrix} y_1 - h(x_1) \\ y_2 - h(x_2) \\ \vdots \\ y_i - h(x_i) \\ \vdots \\ y_n - h(x_n) \end{vmatrix} & = & \begin{vmatrix} e_1 \\ e_2 \\ \vdots \\ e_i \\ \vdots \\ e_n \end{vmatrix}
\end{array}
$$

$$[e_1 e_2 \cdots e_i \cdots e_n] \cdot \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_i \\ \vdots \\ e_n \end{bmatrix} = \sum_{i=0}^{n} e_i^2 = e^T \cdot e$$

$$\mathcal{L} = \frac{e^t . e}{2n}$$

$$\mathcal{L} = \frac{||Y - h(X)||_2^2}{2n}$$

Normal L2

```python
def train(x, y, umbral, alfa, p):
    np.random.seed(2001)
    w = [np.random.rand() for i in range(1,p+1)]
    y_pred = h(x,w)
    L = Error(y, y_pred)
    while (L > umbral):
        dw = derivada(x, y, w)
        w = update(w,db, alfa)
        y = h(x,w)
        L = Error(y, y_pred)
    return w,L
```

```python
def train(x, y, umbral, alfa, p):
    np.random.seed(2001)
    w = [np.random.rand() for i in range(1,p+1)]
    y_pred = h(x,w)
    L = Error(y, y_pred)
    while (L > umbral):
        dw = derivada(x, y, w)
        w = update(w,db, alfa)
        y = h(x,w)
        L = Error(y, y_pred)
    return w,L
```

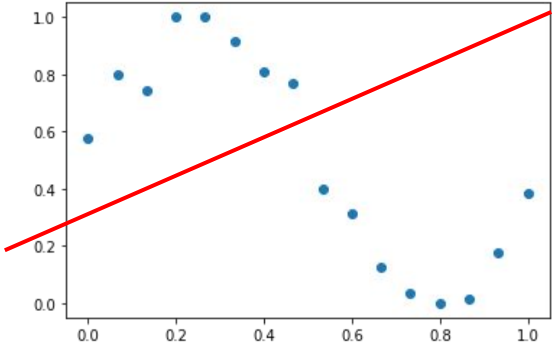
```
def train(x, y, umbral, alfa, p):
    np.random.seed(2001)
    w = [np.random.rand() for i in range(1,p+1)]
    y_pred = h(x,w)
    L = Error(y, y_pred)
    while (L > umbral):
        dw = derivada(x, y, w)
        w = update(w,db, alfa)
        y = h(x,w)
        L = Error(y, y_pred)
    return w,L
```

```python
def train(x, y, umbral, alfa, p):
    np.random.seed(2001)
    w = [np.random.rand() for i in range(1,p+1)]
    y_pred = h(x,w)
    L = Error(y, y_pred)
    while (L > umbral):
        dw = derivada(x, y, w)
        w = update(w,db, alfa)
        y = h(x,w)
        L = Error(y, y_pred)
    return w,L
```

```python
def train(x, y, umbral, alfa, p):
    np.random.seed(2001)
    w = [np.random.rand() for i in range(1,p+1)]
    y_pred = h(x,w)
    L = Error(y, y_pred)
    while (L > umbral):
        dw = derivada(x, y, w)
        w = update(w,db, alfa)
        y = h(x,w)
        L = Error(y, y_pred)
    return w,L
```

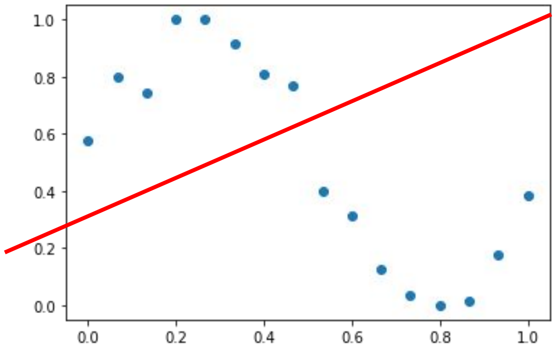
2 Regularización

Underfitting



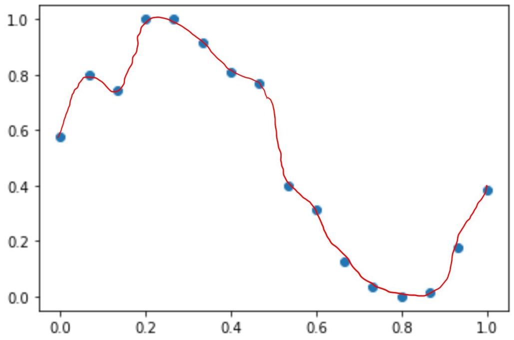
$$h(x_i) = x_i w + b$$

Underfitting



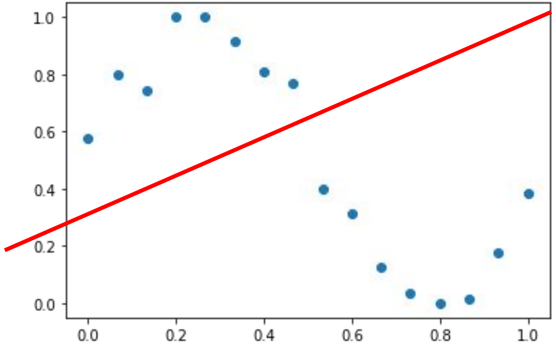
$$h(x_i) = x_i w + b$$

Overfitting



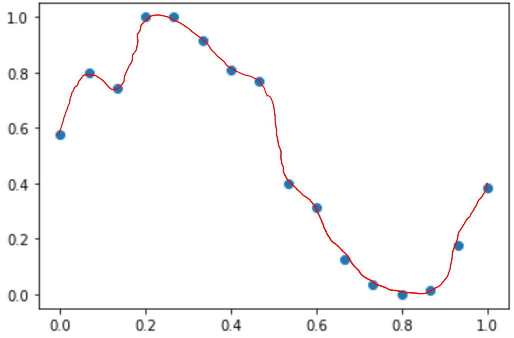
$$h(x_i) = x_i^0 w_0 + x_1^1 w_1 + \ldots + x_i^{20} w_{20}$$

Underfitting



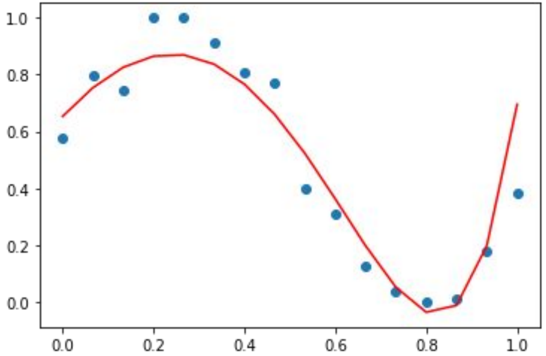
$$h(x_i) = x_i w + b$$

Overfitting



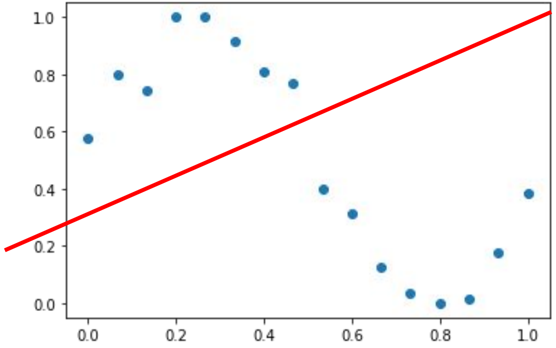
$$h(x_i) = x_i^0 w_0 + x_1^1 w_1 + ... + x_i^{20} w_{20}$$

good



$$h(x_i) = x_i^0 w_0 + x_1^1 w_1 + ... + x_i^3 w_3$$

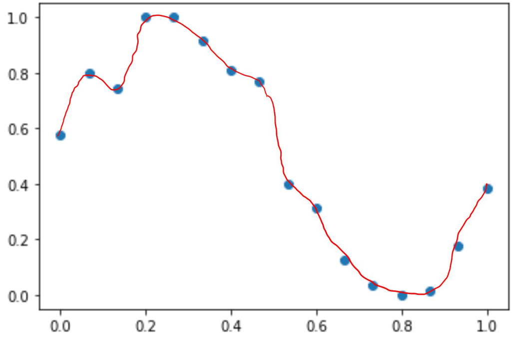
Underfitting



$$h(x_i) = x_i w + b$$

- Modelo simple
- Modelo con poca capacidad

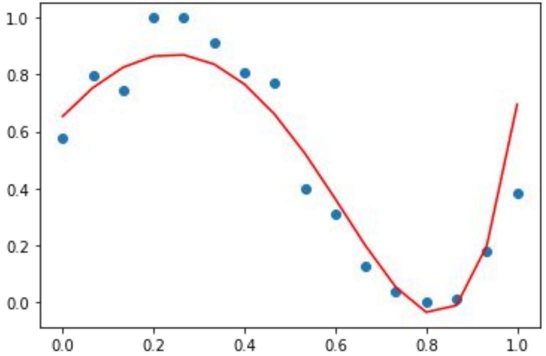
Overfitting



$$h(x_i) = x_i^0 w_0 + x_1^1 w_1 + ... + x_i^{20} w_{20}$$

- Modelo muy complejo
- Modelo con mucha capacidad

good



$$h(x_i) = x_i^0 w_0 + x_1^1 w_1 + ... + x_i^3 w_3$$

- Modelo que se ajusta a los datos
- Modelo con capacidad adecuada

¿Cómo hacemos para disminuir la complejidad de nuestro modelo?

Regularización

Término regularizador

$$\mathcal{L} = \frac{||Y - h(X)||_2^2}{2n} + \boxed{\mathcal{R}(w)}$$

$$\mathcal{L} = \frac{\sum_{i=0}^{n}(y_i - h(x_i))^2}{2n} + \boxed{\frac{\lambda}{n}\sum_{j=1}^{p} w_j^2}$$

$$\mathcal{L} = min\{\frac{\sum_{i=0}^{n}(y_i - h(x_i))^2}{2n} + \frac{\lambda}{n}\sum_{j=1}^{p} w_j^2\}$$

$$\mathcal{L} = \frac{\sum_{i=0}^{n}(y_i - h(x_i))^2}{2n} + \boxed{\lambda \sum_{j=1}^{p} w_j^2}$$

$$\mathcal{L} = min\{ \frac{\sum_{i=0}^{n}(y_i - h(x_i))^2}{2n} + \lambda \sum_{j=1}^{p} w_j^2 \ \}$$

$$\boxed{\mathcal{L} = \frac{||Y - XW^t||_2^2}{2n} + \lambda ||W||_2^2}$$

¿Qué ocurre si el parámetro λ es muy grande ?

λ = 10000

$$\mathcal{L} = \frac{||Y - XW^t||_2^2}{2n} + \lambda||W||_2^2$$

¿Qué ocurre si el parámetro λ es muy pequeño ?

λ = 0.01

$$\mathcal{L} = \frac{||Y - XW^t||_2^2}{2n} + \lambda||W||_2^2$$

¿A quién afecta o quienes cambian si modificamos la función de pérdida?

Las Derivadas

$$\frac{\partial \mathcal{L}}{\partial w_i} = \frac{\partial(\frac{||Y - XW^t||_2^2}{2n} + \lambda|W||_2^2)}{\partial w_i}$$

Hipótesis $$h(X) = X * w^T$$

Loss $$\mathcal{L} = \frac{||Y - XW^t||_2^2}{2n} + \boxed{\frac{\lambda}{n}||W||_2^2}$$

Derivadas $$\frac{\partial \mathcal{L}}{\partial w_j} = \frac{\sum_{i=0}^{n}(y_i - h(x_i)) * (-x_i^j)}{n} + \boxed{\frac{2\lambda w_j}{n}}$$

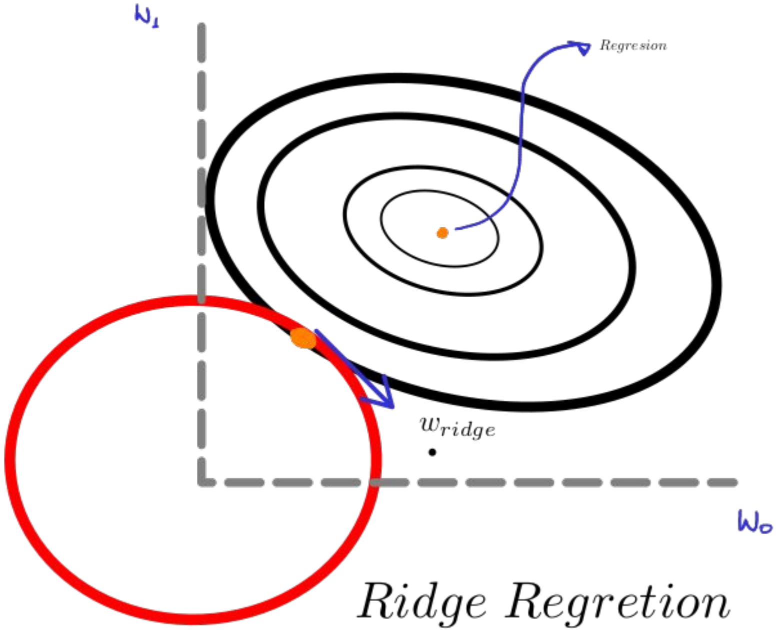
2.1 Métodos de Regularización en Regresión

Ridge

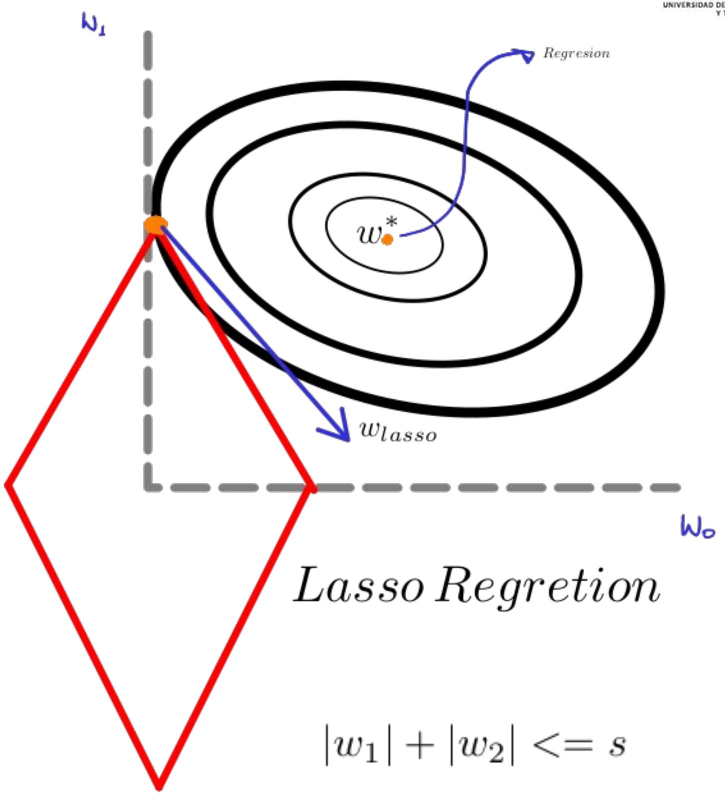
Loss $$\mathcal{L} = \frac{||Y - XW^t||_2^2}{2n} + \boxed{\lambda||W||_2^2}$$

Lasso

Loss $$\mathcal{L} = \frac{||Y - XW^t||_2^2}{2n} + \boxed{\lambda||W||_1}$$



$$w_1^2 + w_2^2 <= s$$



$$|w_1| + |w_2| <= s$$

$$\mathcal{L} = \frac{||Y - XW^t||_2^2}{2n} + \lambda||W||_2^2$$

$$\cup$$

$$\mathcal{L} = \frac{||Y - XW^t||_2^2}{2n} + \lambda||W||_1$$

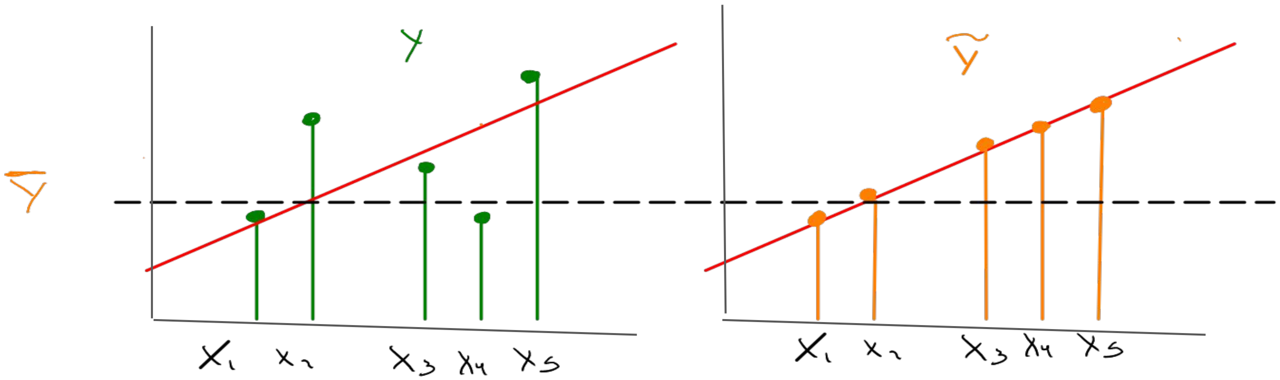
Valores pequeños de W

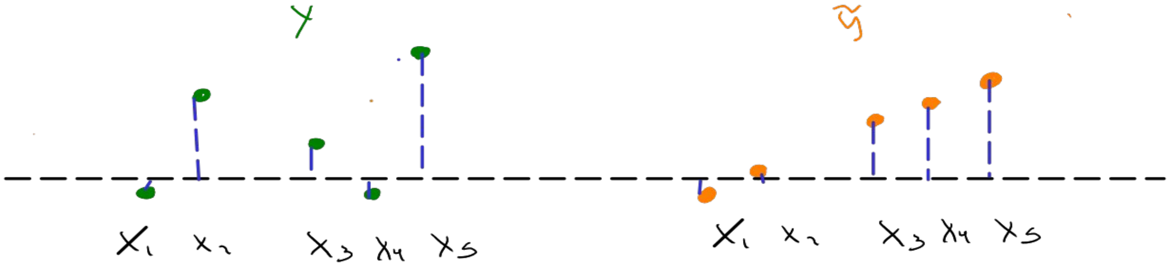
Vector con varios valores cero

Elastinet

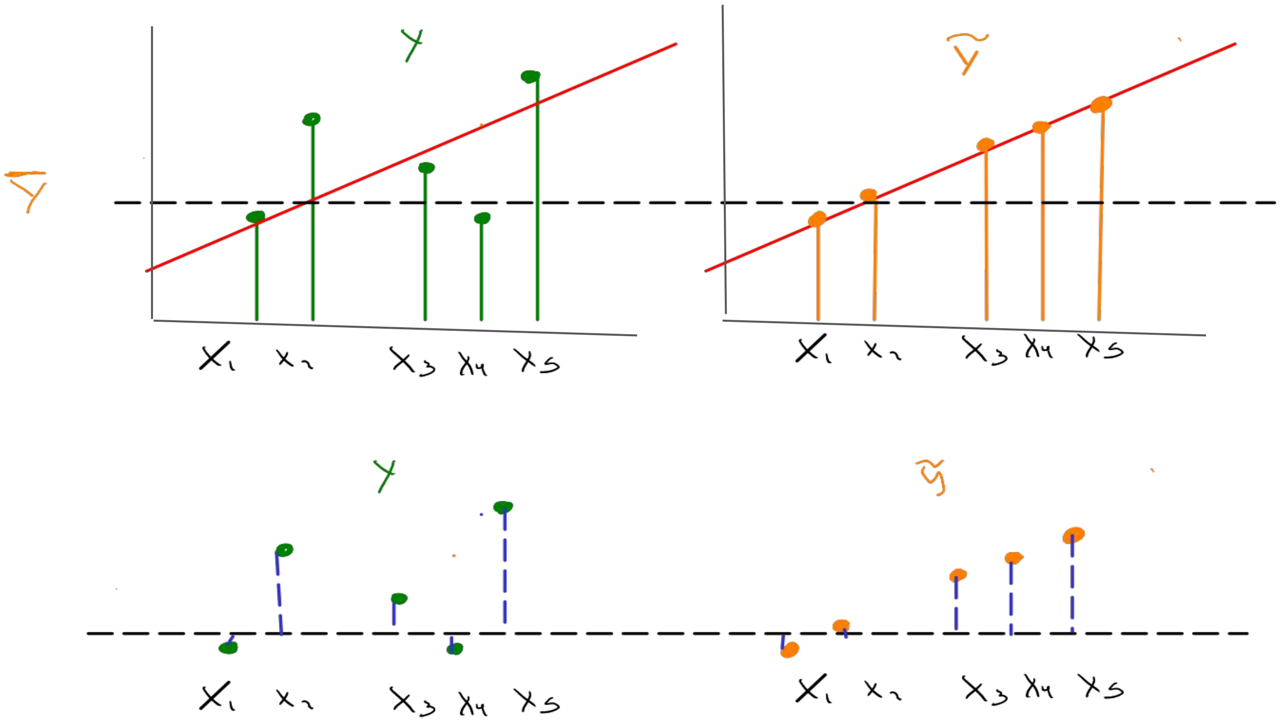
$$\mathcal{L} = \frac{||Y - XW^t||_2^2}{2n} + \rho\lambda||W||_1 + (1-\rho)\lambda||W||_2$$

2.3 Medida de calidad de la regresión


$\bar{Y}$
Y
X_1 X_2 X_3 X_4 X_5
$\tilde{Y}$
X_1 X_2 X_3 X_4 X_5


Y
X_1 X_2 X_3 X_4 X_5
$\tilde{y}$
X_1 X_2 X_3 X_4 X_5

UTEC UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA



Coeficiente de determinación

$$R^2 = \frac{\sum_{i=1}^{n} (\hat{y}_i - \overline{Y})^2}{\sum_{i=1}^{n} (y_i - \overline{Y})^2}$$

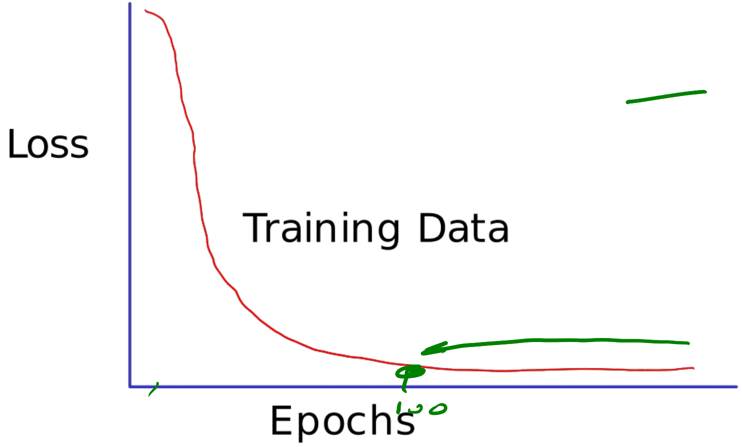
3 Entrenamiento, bias y varianza

Data Set	
Training data	Testing data
80%	20%

loss, w = training(x_train, y_traing, alpha, epochs)

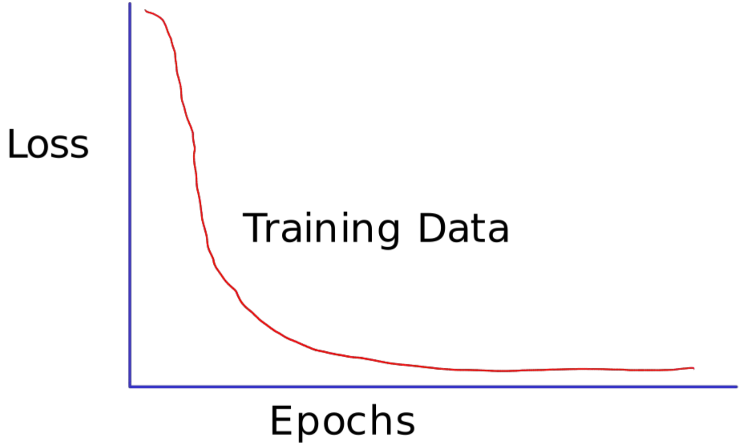
while (E > umbr)

for (i=1; i< Epocas



Data Set		
Training data	Val	Test
70%	20%	10%

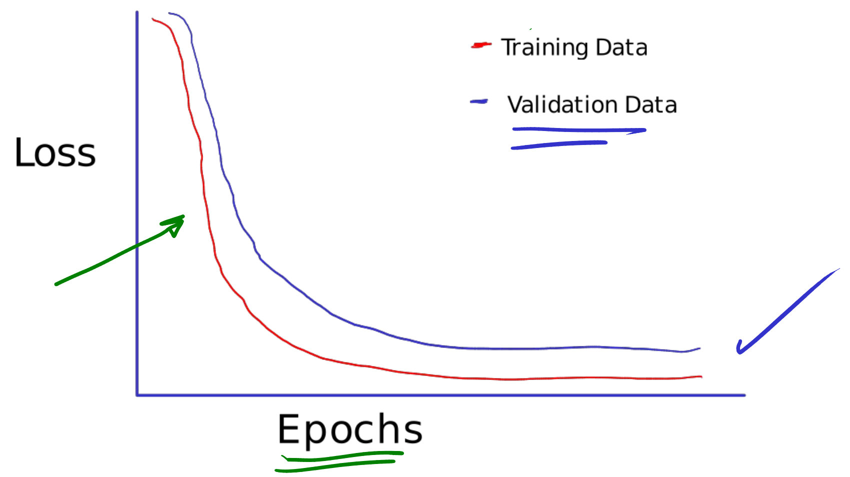
loss, w = training(x_train, y_traing, alpha, epochs)



Data Set

Training data	Val (dev)	Test
70%	20%	10%

loss, w = training(x_train, y_traing, alpha, epochs)



```
traing(x_train, y_train, alpha, epochs):
  w = # Initialize parameters
  for i in range(epochs):
    x,y  =  get_data(x_train, y_train, batch)
    dw = derivatives(x,y,w)
    w   = Change_Parameters(w,dw, alpha)
    loss_training = Error(x,y,w,batch)
    loss_val = Error(x_val,y_val,w,batch)
    L_T.append(loss_training)
    L_V.append(loss_val)
  return L_T, L_V, w
```

Todos

- sd
- sgd
- MB

uno a uno

64

10^9

10^9

gd

10^6

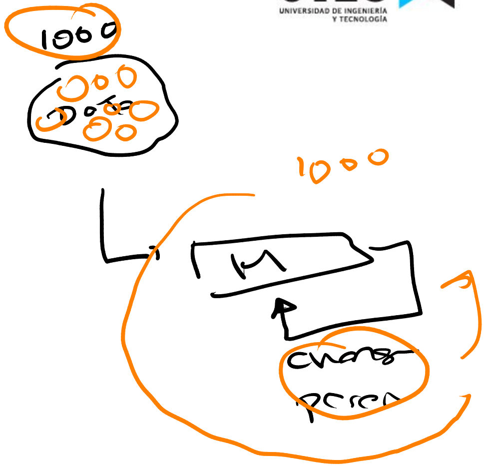
$\sum_{i=0}^{n} \frac{(y_i - \hat{y}_i)^2}{2n}$

gradient descendant

stocastic gradient descendant

mini batch

```
traing(x_train, y_train, alpha, epochs):
  w = # Initialize parameters
  for i in range(epochs):
     x,y  =  get_data(x_train, y_train, batch)
     dw = derivatives(x,y,w)
     w   = Change_Parameters(w,dw, alpha)
     loss_training = Error(x,y,w,batch)
     loss_val = Error(x_val,y_val,w,batch)
     L_T.append(loss_training)
     L_V.append(loss_val)
  return L_T, L_V, w
```



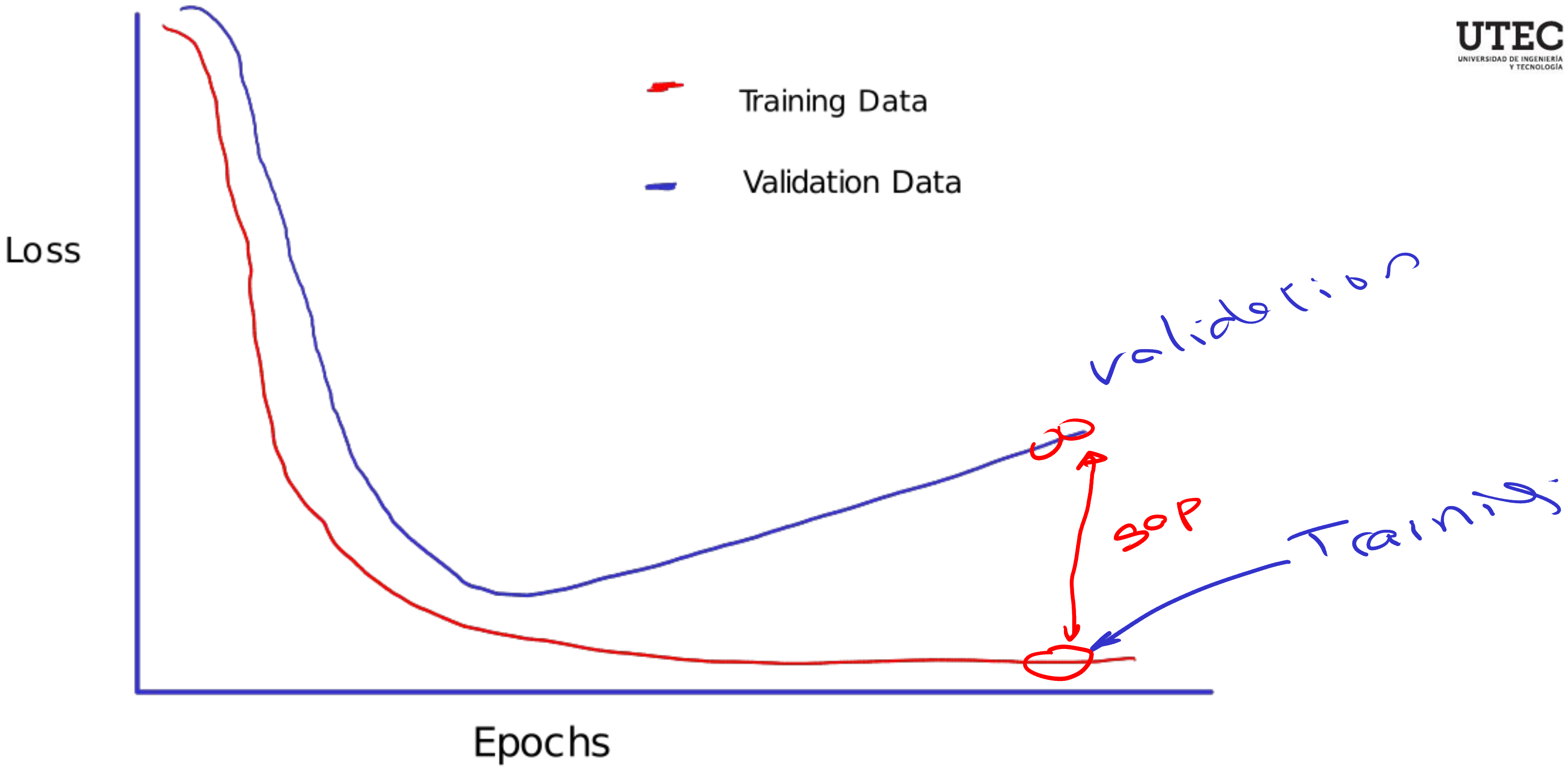
SD → Lento

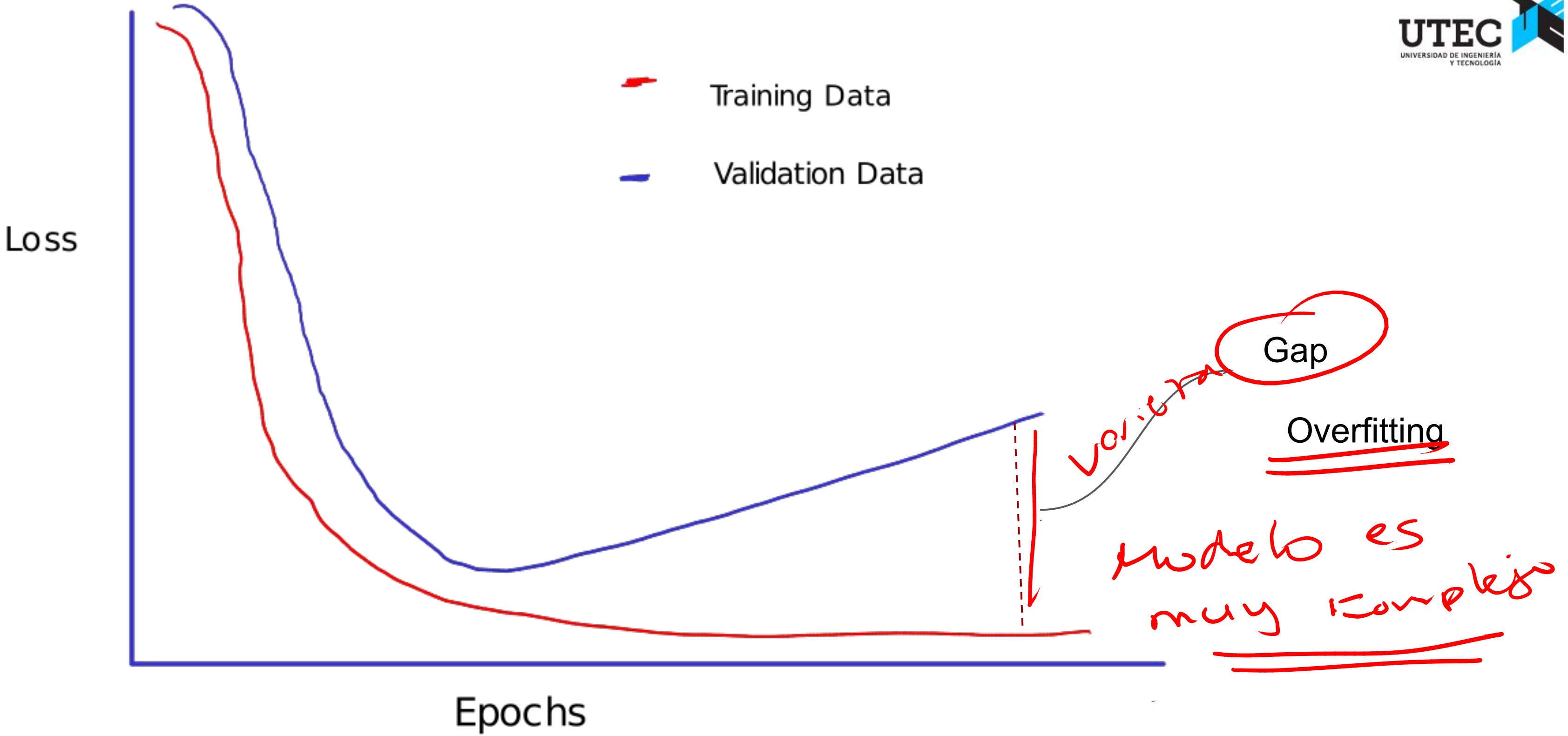
SGD →

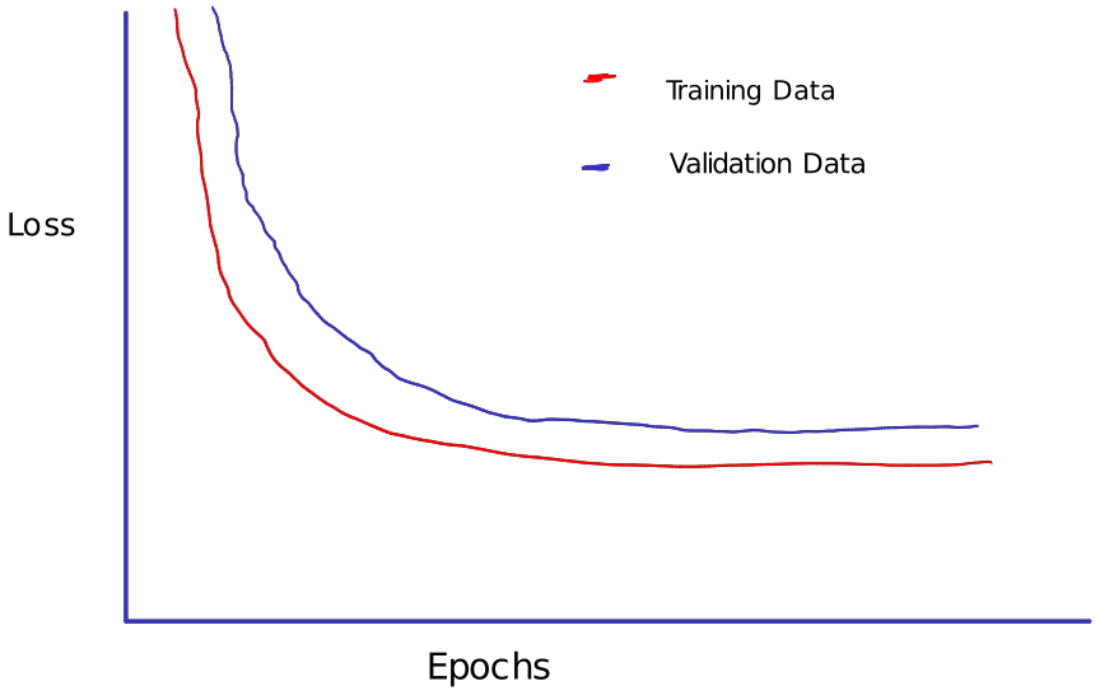
```
traing(x_train, y_train, alpha, epochs):
  w = # Initialize parameters
  for i in range(epochs):
     x,y  =  get_data(x_train, y_train, batch)
     dw = derivatives(x,y,w)
     w   = Change_Parameters(w,dw, alpha)
     loss_training = Error(x,y,w,batch)
     loss_val = Error(x_val,y_val,w,batch)
     L_T.append(loss_training)
     L_V.append(loss_val)
  return L_T, L_V, w
```

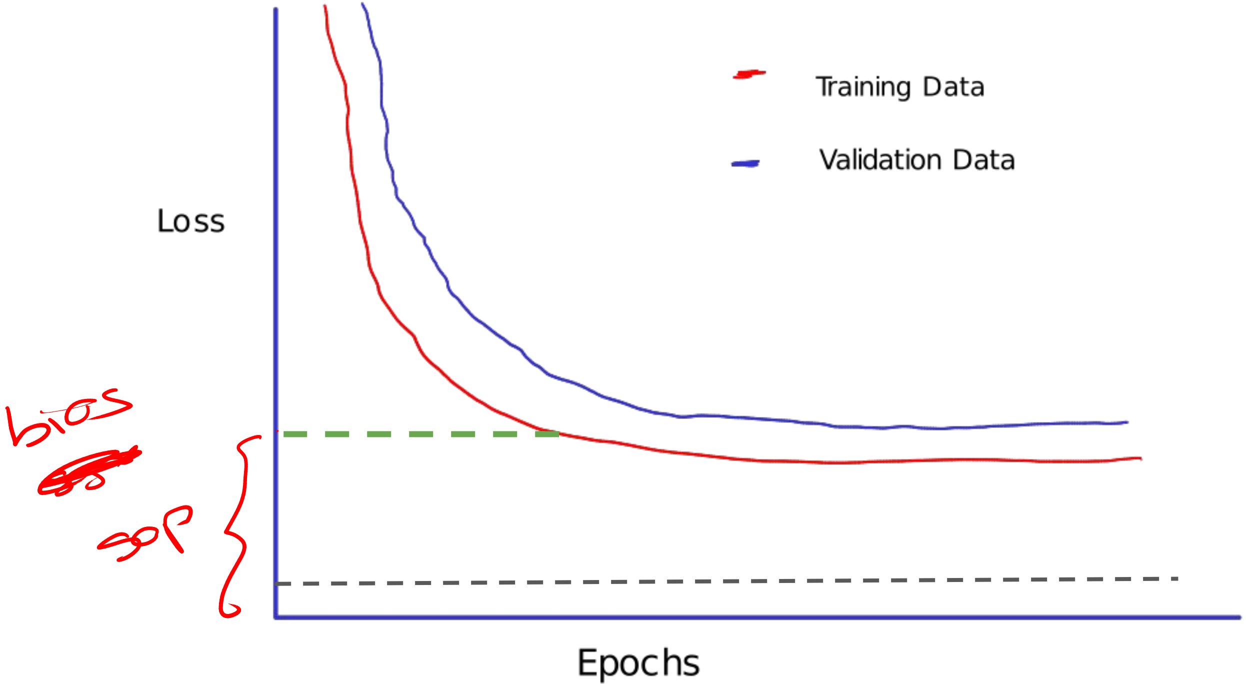
```
traing(x_train, y_train, alpha, epochs):
  w = # Initialize parameters
  for i in range(epochs):
      x,y  =  get_data(x_train, y_train, batch)
      dw = derivatives(x,y,w)
      w   = Change_Parameters(w,dw, alpha)
      loss_training = Error(x,y,w,batch)
      loss_val = Error(x_val,y_val,w,batch)
      L_T.append(loss_training)
      L_V.append(loss_val)
  return L_T, L_V, w


  plot(loss_t, loss_v, epochs)
```

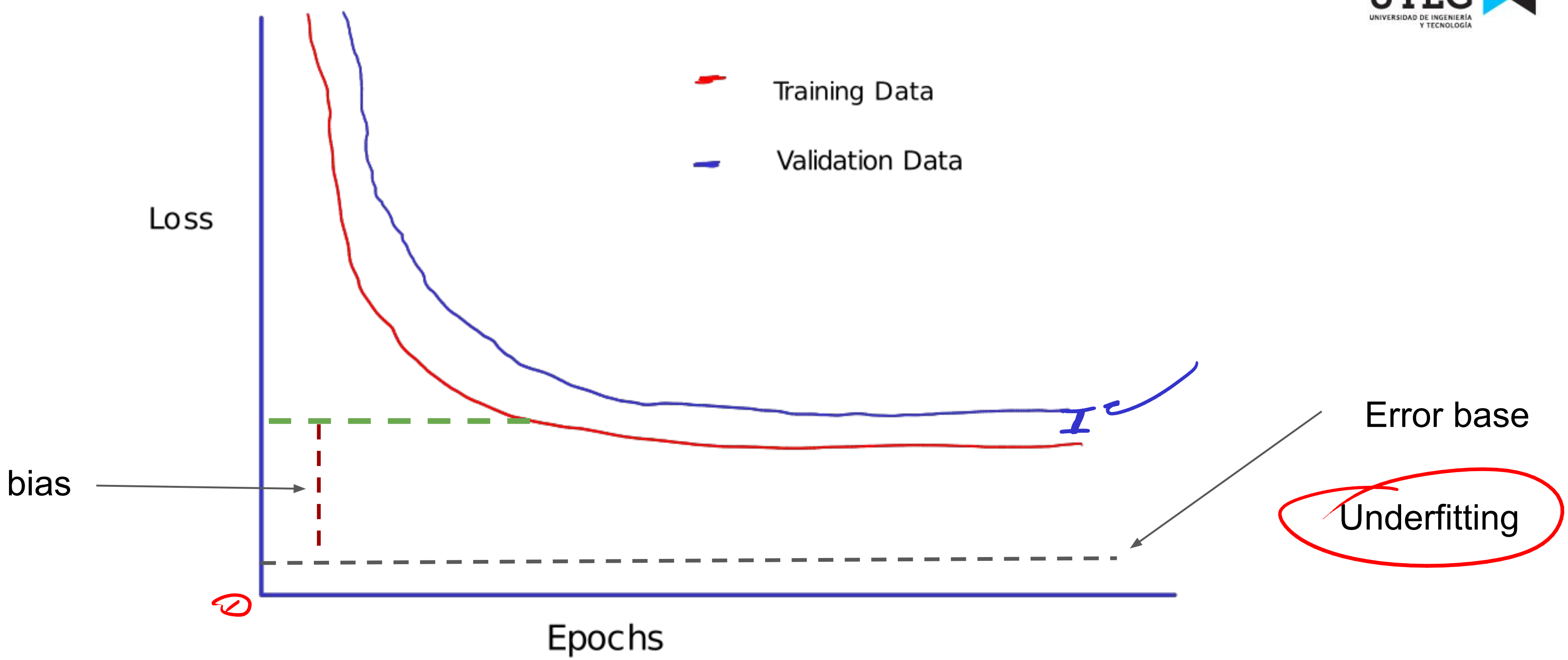
```
traing(x_train, y_train, alpha, epochs):
  w = # Initialize parameters
  for i in range(epochs):
      x,y  =  get_data(x_train, y_train, batch)
      dw = derivatives(x,y,w)
      w   = Change_Parameters(w,dw, alpha)
      loss_training = Error(x,y,w,batch)
      loss_val = Error(x_val,y_val,w,batch)
      L_T.append(loss_training)
      L_V.append(loss_val)
  return L_T, L_V, w


  plot(loss_t, loss_v, epochs)
```


Training Data
Validation Data
Loss
Epochs
validation
Training

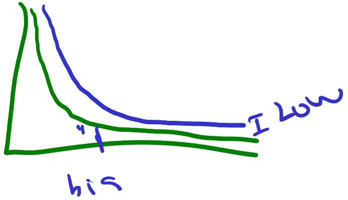

Training Data
Validation Data
Loss
Epochs
Gap
Overfitting
varianza
modelo es
muy complejo


Training Data
Validation Data
Loss
Epochs

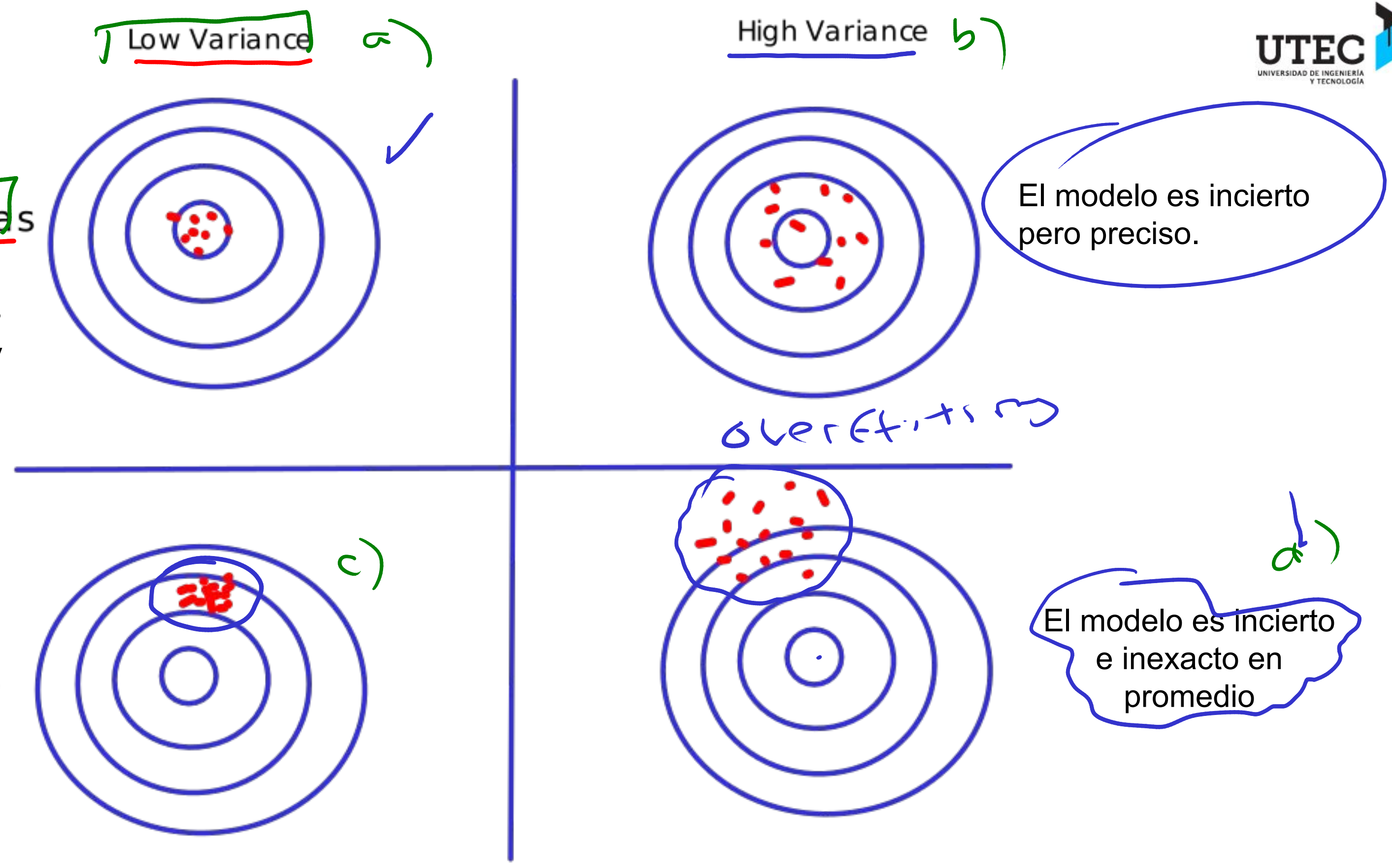

Training Data
Validation Data
Loss
Epochs
bias
gap



Training Data
Validation Data
Loss
bias
Error base
Underfitting
Epochs

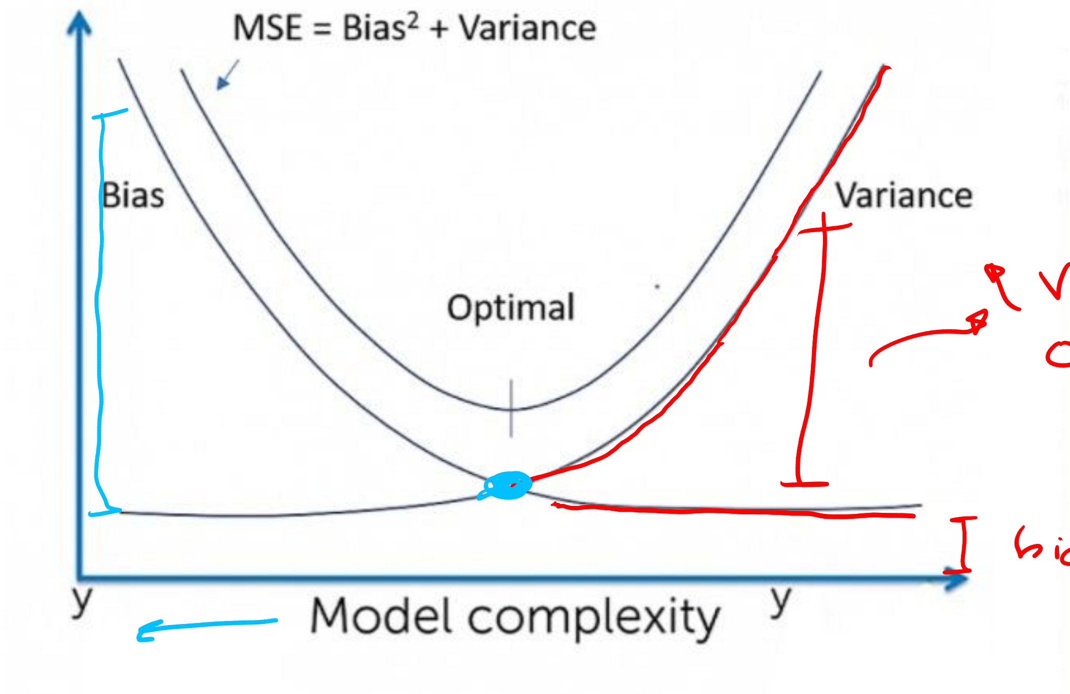


I Low
bis



Low Variance
a)
High Variance
b)
Low bias
El modelo es consistente y preciso
El modelo es incierto pero preciso.
overfitting
c)
d)
High Bias
El modelo es consistente pero inexacto.
El modelo es incierto e inexacto en promedio

Loss



varianza
overfitting

bias

Fuente: https://editor.analyticsvidhya.com/uploads/983161.png

Clasificación

Cristian López Del Alamo
clopezd@utec.edu.pe
IPRODAM3D - Research group

2022

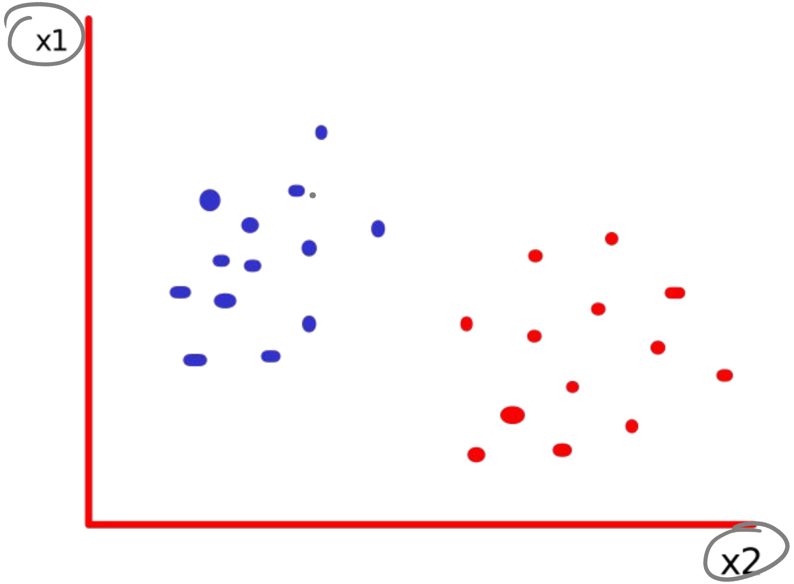
Clasificación

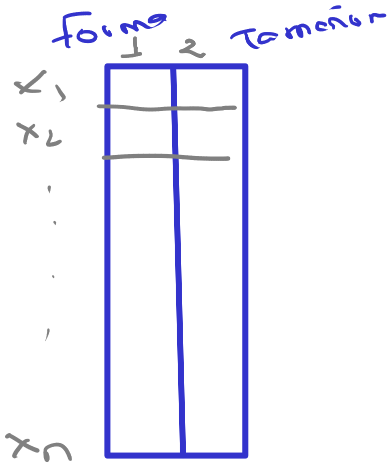
Supervi

(x, y)

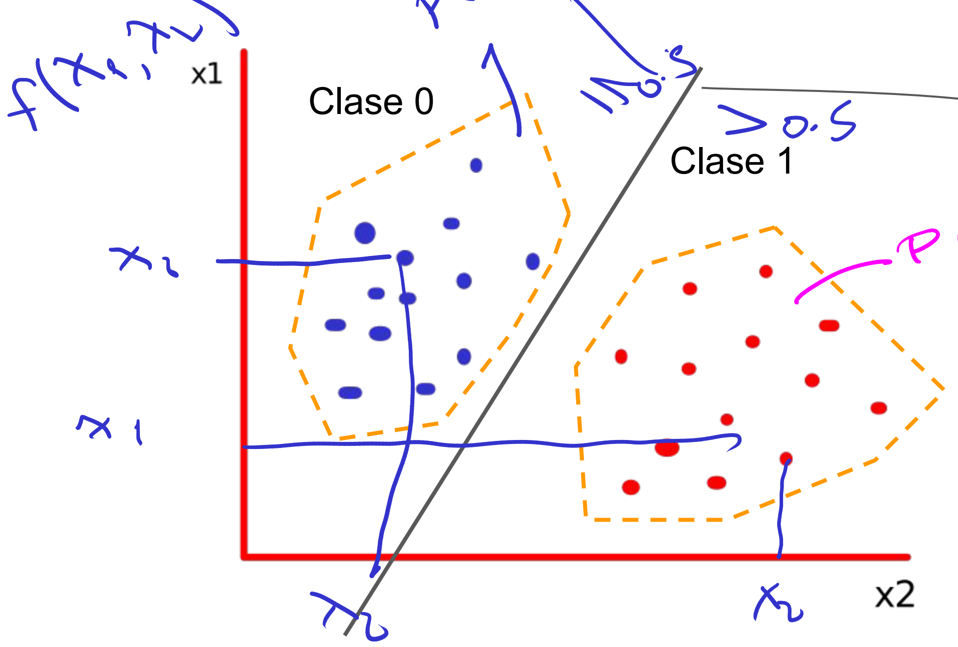
x

$y - y$


x1
x2


forma
tamaño
1
2
x1
x2
xn

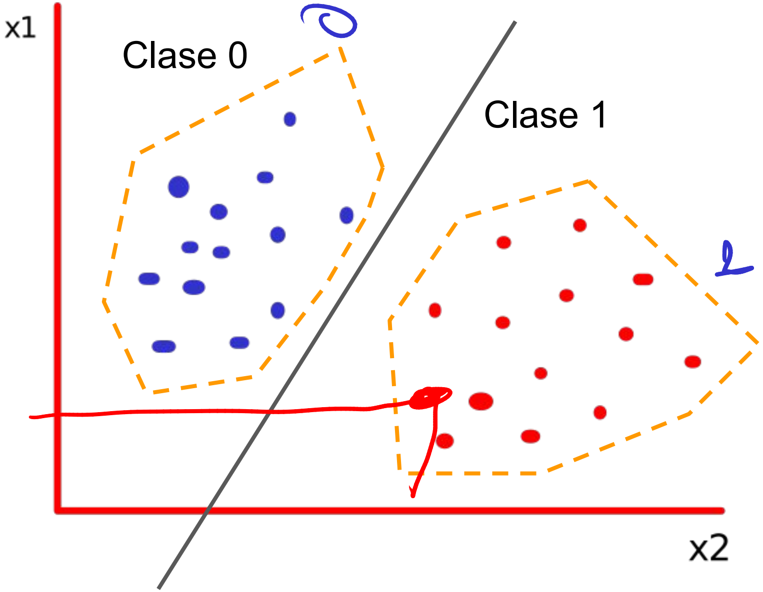
Clasificación Binaria



$$h(x_i) = w_0 + x_i w_1$$

$f(x_1, x_2) = 0.9$

Clasificación Binaria



$$h(x_i) = w_0 + x_1 w_1$$

$$s(x_i) = \frac{1}{1 + e^{-h(x_i)}}$$

$S(x_1, x_2) = 0.2$

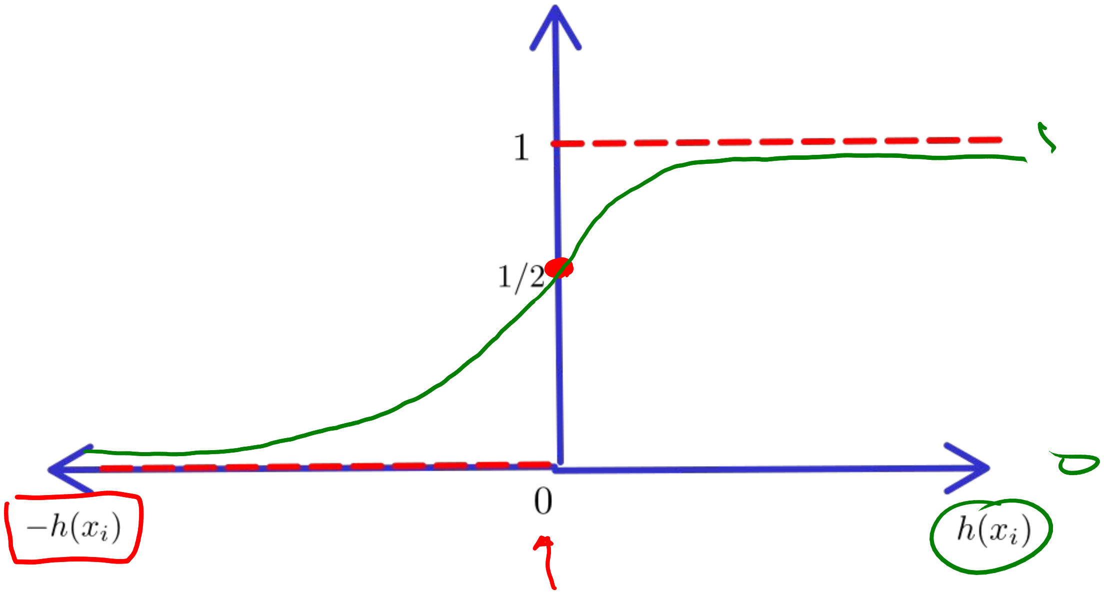
$S(x_1, x_2) = 0.85$

Clasificación Binaria

$x_i = \langle x_{i1}, x_{i2} \rangle$

$s(x_i)$

$$s(x_i) = \frac{1}{1 + e^{\boxed{-h(x_i)}}}$$



$$\frac{1}{1 + \frac{1}{e^0}}$$

$$\frac{1}{1 + e^{-(-\infty)}}$$

$$\frac{1}{\infty}$$

Hipótesis $$s(x_i) = \frac{1}{1+e^{-h(x_i)}}$$

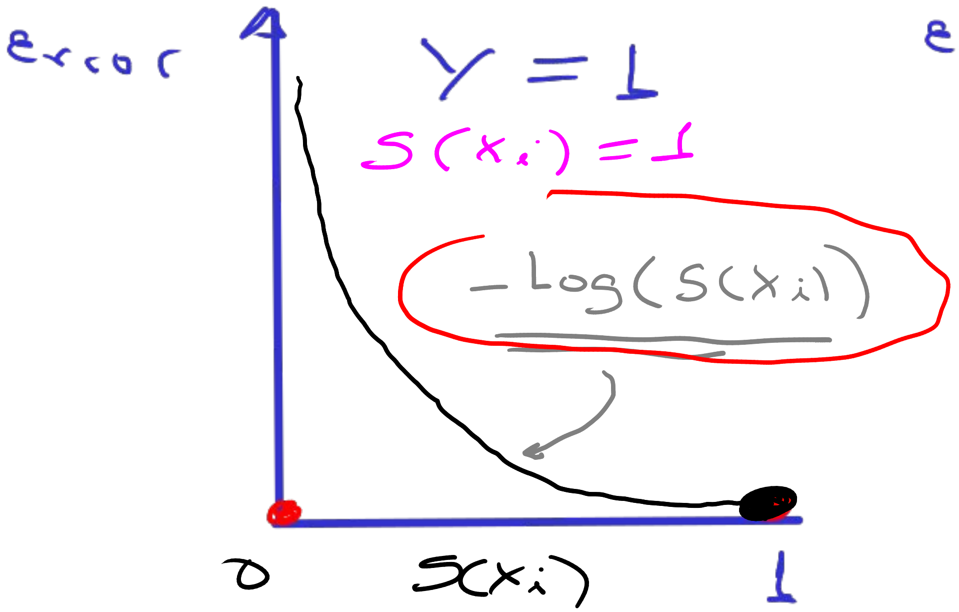
$s(x_i) = [0, 1]$

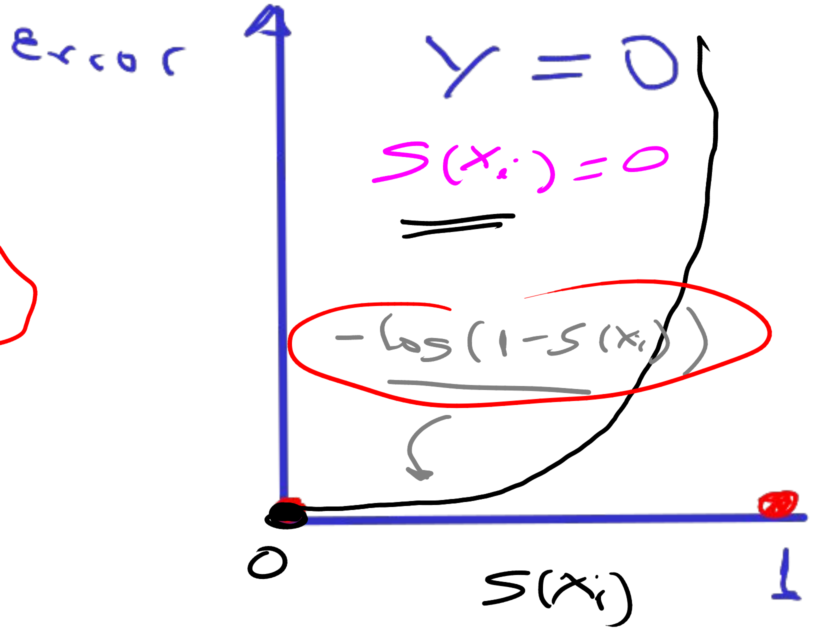
$y_i = 1 \quad s(x_i) = 0.1$

Loss $$\mathcal{L} = -\sum_{i=1}^{n} \underbrace{\cancel{\left(y_i \log(s(x_i))\right)}}_{1} + \underbrace{\cancel{\left((1-y_i)\log(1-s(x_i))\right)}}_{2}$$

1 $y = 1$

2 $y = 0$


error
Y = 1
S(xi) = 1
-Log(S(xi))
0
S(xi)
1


error
Y = 0
S(xi) = 0
-log(1-S(xi))
0
S(xi)
1

Hipótesis $s(x_i) = \dfrac{1}{1 + e^{-h(x_i)}}$

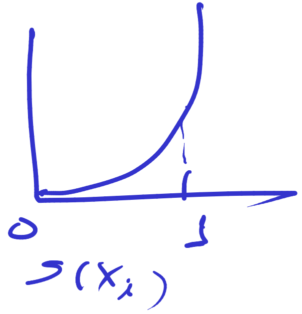
Intuición

$y = 1$

$s(x_i) = 0.7$

$y = 0$

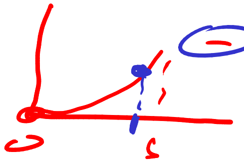
$s(x_i) = 0.9$



Loss $\mathcal{L} = -\sum_{i=1}^{n} (y_i \log(s(x_i)) + (1 - y_i) \log(1 - s(x_i)))$

$(-\log(s(x_i)))$

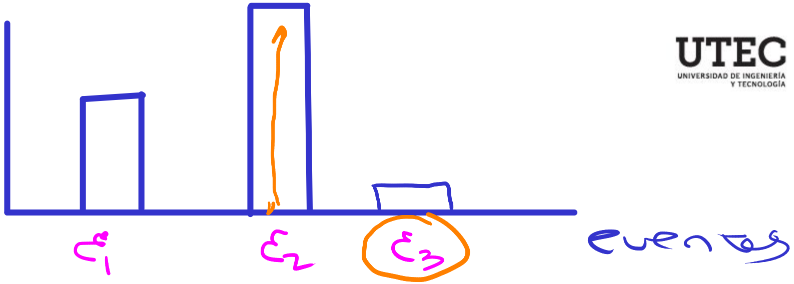
$-\log(1 - s(x_i))$



Derivadas $\dfrac{\partial L}{w_j} = \dfrac{1}{n} \sum_{i=1}^{n} (y_i - s(x_i))(-x_{ij})$

$\boxed{-\log(P(x_i))}$

$e = 0.0001$ $s(x_i) + e$ frec



más información

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

$$\boxed{\text{Hipótesis} \qquad s(x_i) = \frac{1}{1 + e^{-h(x_i)}}}$$

$s(x_i) + e$

$y = 1$ $\qquad$ $y = 0$

$$\boxed{\text{Loss} \qquad \mathcal{L} = -\sum_{i=1}^{n}(y_i \log(s(x_i)) + (1 - y_i)\log(1 - s(x_i)))}$$

$$\text{Derivadas} \qquad \frac{\partial L}{w_j} = \frac{1}{n}\sum_{i=1}^{n}(y_i - s(x_i))(-x_{ij})$$

logística ??

$$\log_2\left(\frac{1}{2^{-50}}\right)$$

$$\log\left(\frac{1}{P(x)}\right) \qquad \log 1$$

$$\boxed{-\log(P(x))}$$

Hipótesis $$s(x_i) = \frac{1}{1 + e^{-h(x_i)}}$$

Loss $$\mathcal{L} = -\sum_{i=1}^{n} (y_i \log(s(x_i)) + (1 - y_i) \log(1 - s(x_i)))$$

Derivadas $$\frac{\partial L}{w_j} = \frac{1}{n} \sum_{i=1}^{n} (y_i - s(x_i))(-x_{ij})$$

Hipótesis $$s(x_i) = \frac{1}{1 + e^{-h(x_i)}}$$

Loss $$\mathcal{L} = -\sum_{i=1}^{n}(y_i \log(s(x_i)) + (1 - y_i)\log(1 - s(x_i)))$$


INGENIERÍA MECATRÓNICA
BIOINGENIERÍA
CIENCIA DE LA COMPUTACIÓN
UTEC
UNIVERSIDAD DE INGENIERÍA
Y TECNOLOGÍA